



# Finding Performance and Power Issues on Android Systems

By Eric W Moore



# Agenda

Performance & Power Tuning on Android

& Features Needed/Wanted in a tool

Some Performance Tools

Getting a Device that Supports the Tools

Analyzing a CPU App Micro-Benchmark

Finding a Device Driver Issue

Examining an Android Game

Summary

# Performance Tuning

1. Determine performance requirement
2. Get current reasonably optimized benchmark/system – that supports the tools – and establish baseline performance
3. Find code that consumes the most time (hotspot)
  - (Optional) Find the efficiency of that code
4. Optimize the hotspot
  - Use Lowest Effort / Highest Effect Optimization
    - Remove
    - Algorithm Change
    - Micro-Architectural Optimization
    - Hardware Upgrade
    - OS Tuning
5. If perf requirement not met - Go to Step 3.

# Needed/Wanted Performance Tool Capabilities

## 1. Measure & report everything..

- Hotspot
  - Various metrics
    - CPU
    - GPU
    - Other SoC Components
  - System wide
  - Memory Usage
  - Drill to Source
  - Call Stack
  - Call/Loop Counts
  - Wait time (Synchronization - thread, I/O, Network, etc)
  - Latency
  - Etc...
- Problem: Observer Affect - The act of measuring affects what is measured

## 2. Low Overhead / Accurate

## 3. Easy to use: Make it easy to collect and identify issues

# Power Tuning

1. Identify power being used

Or... perhaps more actionable analysis...

2. Do performance tuning = race to idle

3. Find code which is not using the most efficient SoC block for execution

4. Find code which does not need to execute...

- Unnecessarily using a device or the CPU
- Unnecessarily waking up a SoC block, CPU, or system
- Unnecessarily keeping a SoC block, CPU, or the system awake

# Needed/Wanted Power Tool Capabilities

## 1. Sleep Analysis

- What is waking h/w? Why?
- What is keeping it awake? Why?

## 2. Utilization Analysis

- What is active? Why is it active?

## 3. Power Analysis

- What is consuming power? How much?

## 4. Platform perspective

- Understand peripheral power consumers
- Understand behaviors driving peripheral power consumers

## 5. Low Overhead / Accurate

## 6. Easy to use: Make it easy to collect and identify issues

## 7. Correlation allowing seeing all Behaviours on System

## 8. Comparison Features to compare different scenarios

# Some Tools from Google\* to help with Android\* Performance Tuning

- Systrace: Analyze app and system processes
- TraceView: Analyze applications Java profile
- Tracer: To analyze OpenGL ES frames
- Many Memory Analysis Tools: Find costly allocations or leaks In memory usage
- Network Traffic tool: Analyze app & system specific network usage
- Hierarchy Viewer: Used to analyze the layout of your UI.
- Android No Response – Crashes app when UI is non-responsive
- Settings: Developer Options on device – Some basic CPU stats
- Lint: Static analysis tool – offers some performance recommendations

# Some tools from Intel® Corporation to help with Android Performance Tuning

Intel® System Studio 2014

Intel® VTune™ Amplifier 2014 for Systems

Intel® Energy Profiler

Intel® System Analyzer for Android

Other Tools...

Intel® Native Development Experience

Intel® Graphics Performance Analyzers

Intel® Platform Analyzer

Intel® Systems Analyzer

Intel® Frame Analyzer

Other Tools...



# Getting a Device for Performance Analysis and Systems Development

Some of the advanced features require a

- Rooted device
- Drivers

New programs to support this:

## **Intel® Mobile Development Kit for Android**

<http://software.intel.com/mdk>

Dell Venue 8 – Engineering Build

## **<http://01.org/android-ia>**

Get the latest images/sources to support Android on current Intel® architecture devices

# Analyzing a CPU Micro-Benchmark

# A CPU Micro-Benchmark - Pi

Calculates Pi via a Monte Carlo simulation

First in C/C++

Then on Java

Prints time to Calculate Pi

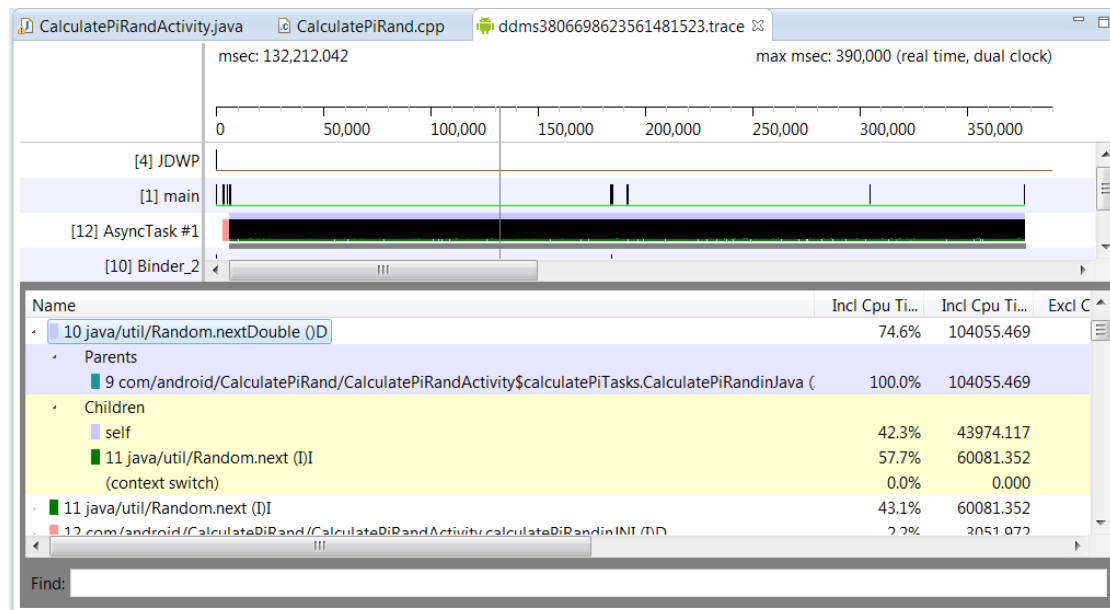
```
for(int i=0; i<(numsteps*2); i++)
    r[i]=rand.nextDouble();
for (int i=0;i<numsteps;i++) {
    double x=r[i];                //X Coordinate
    double y=r[i+BLOCK_SIZE];    //Y Coordinate
    if (x*x + y*y <= 1.0) //Distance from Origin
        dUnderCurve++;    //is under Curve }}
pi= dUnderCurve / (double) numsteps * 4;
```

# Google\* TraceView to Analyze Java\*/Dalvik Apps

- Helps application developers optimize Java source code in an application
- Launched directly from Android's\* Default Eclipse\* Development Environment
- Fairly easy to identify CPU hotspot – based on Java
- Good Java Caller/Callee tree

## Major Issue:

- Observer Effect: The tool disables the JIT – causing wide variances in what you see as performance impact of functions, and what really happens when not run with tool.



# Intel® VTune™ Amplifier for Systems Performance Profiler

## Get the Tuning Data You Need

- Low overhead “hotspot” analysis with call stacks
- Advanced analysis for cache, branching, ...

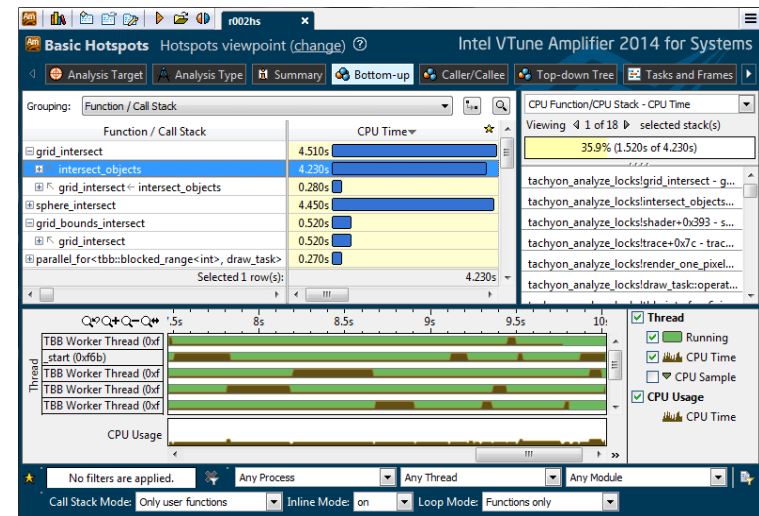
## Find Answers Fast

- Powerful analysis & data mining
- Results mapped to C/C++ or Java source

## Easy to Use

- Remote analysis from the User Interface
- Windows or Linux Host analyzes Linux or Android target

## Available now as part of Intel® System Studio



Optimize Your Software Performance

# Intel VTune Amplifier 2014 for Systems

Easy to get basics working

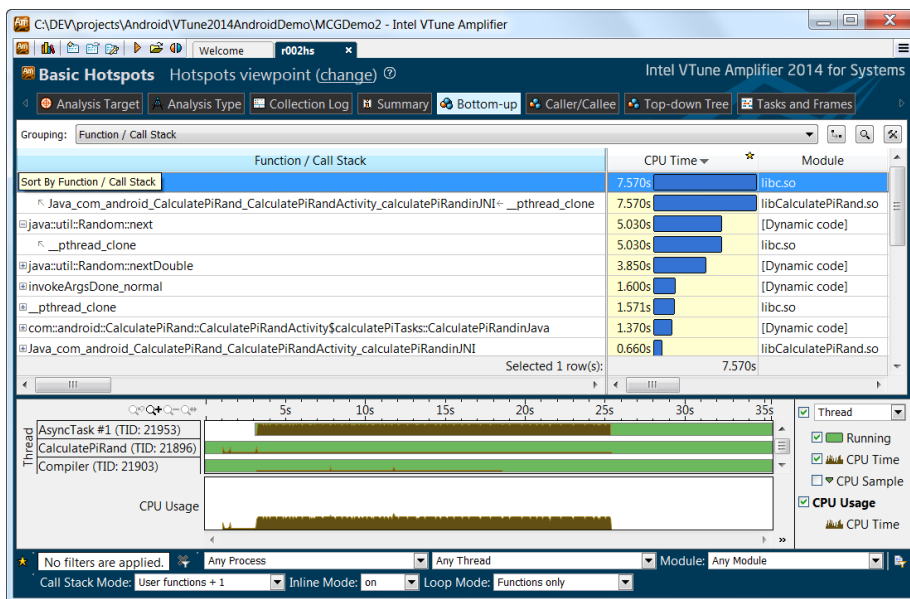
Identification of “Real” Hotspot

Insight into both C/C++ and Java

Issues:

Java(based on Dalvik) and Advanced features requires rooted device with drivers

Java stacks are reported as the “real” stack... not the Java stack.



# Analyzing a Rogue Driver

Intel® Energy Profiler

# Rogue Driver Explained - Driver Initialization

```
static int __init iinit_module( void )
{
    int ret;

    printk("Timer module installing\n");

    // my_timer.function, my_timer.data
    setup_timer( &my_timer, my_timer_callback, 0 );

    printk( "Starting timer to fire in 500ms (%ld)\n", jiffies );
    ret = mod_timer( &my_timer, jiffies + msecs_to_jiffies(500) );
    if (ret) printk("Error in mod_timer\n");

    return 0;
}

static void __exit icleanup_module( void )
{
    int ret;

    ret = del_timer( &my_timer );
    if (ret) printk("The timer is still in use...\n");

    printk("Timer module uninstalling\n");

    return;
}
```

Called when driver is  
loaded thru insmod

Called when driver  
is unloaded thru rmmod



# Rogue Driver Explained - Timer call back

```
void my_timer_callback( unsigned long data )
{
    int i,j,ret;
    for (i=0;i<100000000;i++)
        for (j=0;j<100000000;j++)
            sum=sum+i;
    printk( "my_timer_callback called (%ld). %d\n", jiffies, sum );

    // setup the timer again to fire 500ms
    if (count <50) {
        setup_timer( &my_timer, my_timer_callback, 0 );
        ret = mod_timer( &my_timer, jiffies + msecs_to_jiffies(500) );
        if (ret) printk("Error in my_timer_callback\n");
        count++;
    }
}
```

A busy loop

Timer fires 50 times  
every 500ms

When driver is loaded the a timer fires 50 times every 500ms  
And does some busy work

# Why this Behavior is Important

- Most device drivers operate in this manner
- They get invoked synchronously/asynchronously
- Buried deep in the stack and symptoms typically point to somewhere else

Goal is to run this driver along with workloads of interest and see what effect it has on them and see if our tools can detect this

# Intel® Energy Profiler

Energy and Power Profiler for System Software Developers

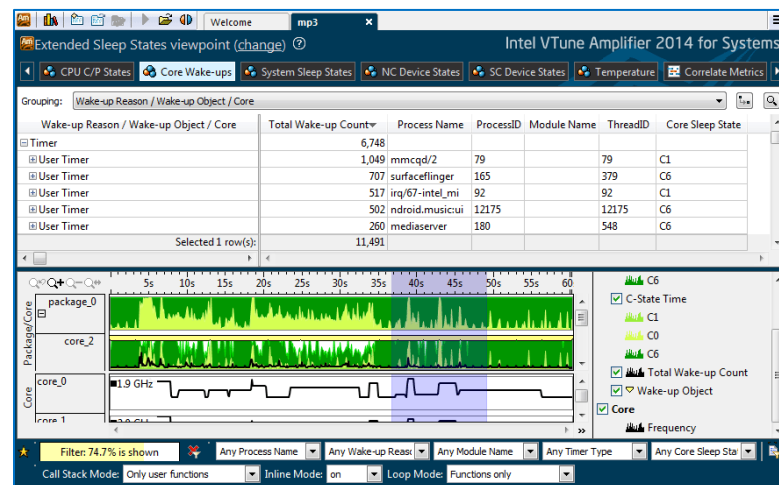
Optimize Software for Extended Battery Life

Find the Cause of Wake Ups That Waste Energy

- Interrupts mapped to the IRQ/device
- Timers mapped to the scheduling process
- Data correlated with Android Wake Locks

Available now for Linux and Android

Part of Intel® System Studio



Requires specific SOCs. On Android, a rootable OS is required with version compatible device drivers. See release notes for details.

Get Actionable Data to Extend Battery Life

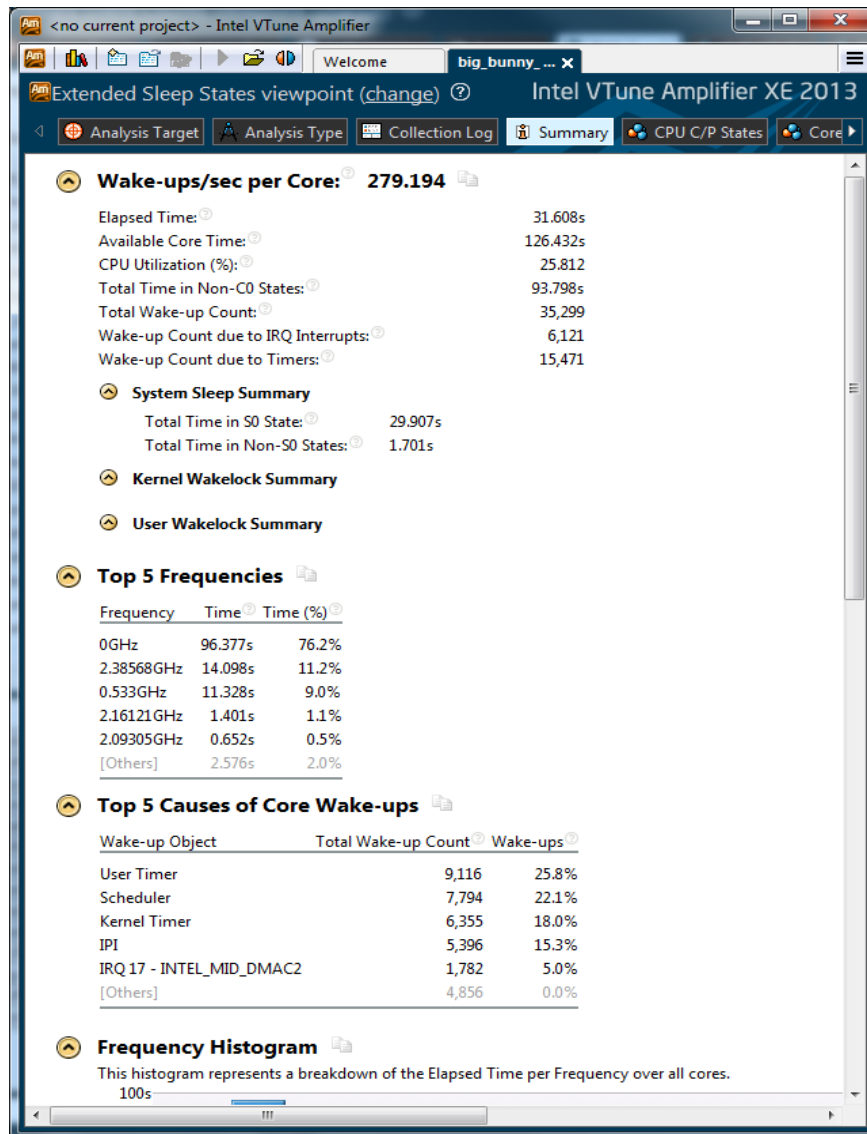
# Video Playback with bad driver

- Get the same video player
- Play a video
- Insert the bad driver
  - `insmod timer_compute.ko`
  - Monitor the framerate (should see a drop)
- Using VTune™ and socwatch to monitor what is happening on the system
  - You can pinpoint the bad driver to be the cause

# Process

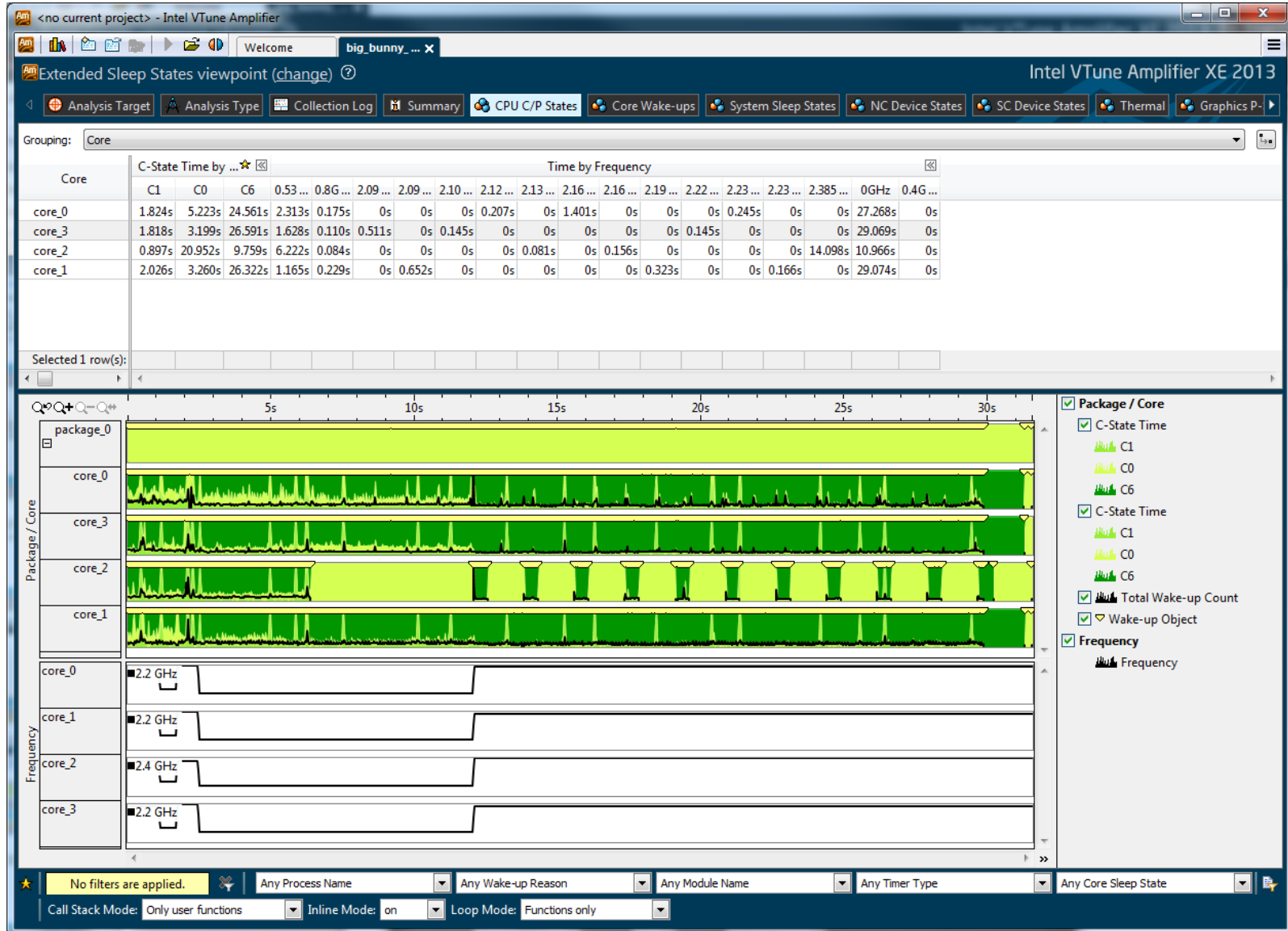
- Play 60 sec of video
- Start Intel Energy Profiler Collection collection
- Insert the timer\_compute.ko
- Import the data and analyze it in VTune™ Amplifier

# Power Analysis via SocWatch



- High wakeups per second
  - 279.194 wakeups/sec
- Significant portion of wakeups are generated by user timers

# Power Analysis via SocWatch



# Analyzing a Game - Need for Speed

Google's Systrace & Tracer

Intel® Graphics Performance Analyzers



# Systrace

Collect easily via the Eclipse or Monitor interface

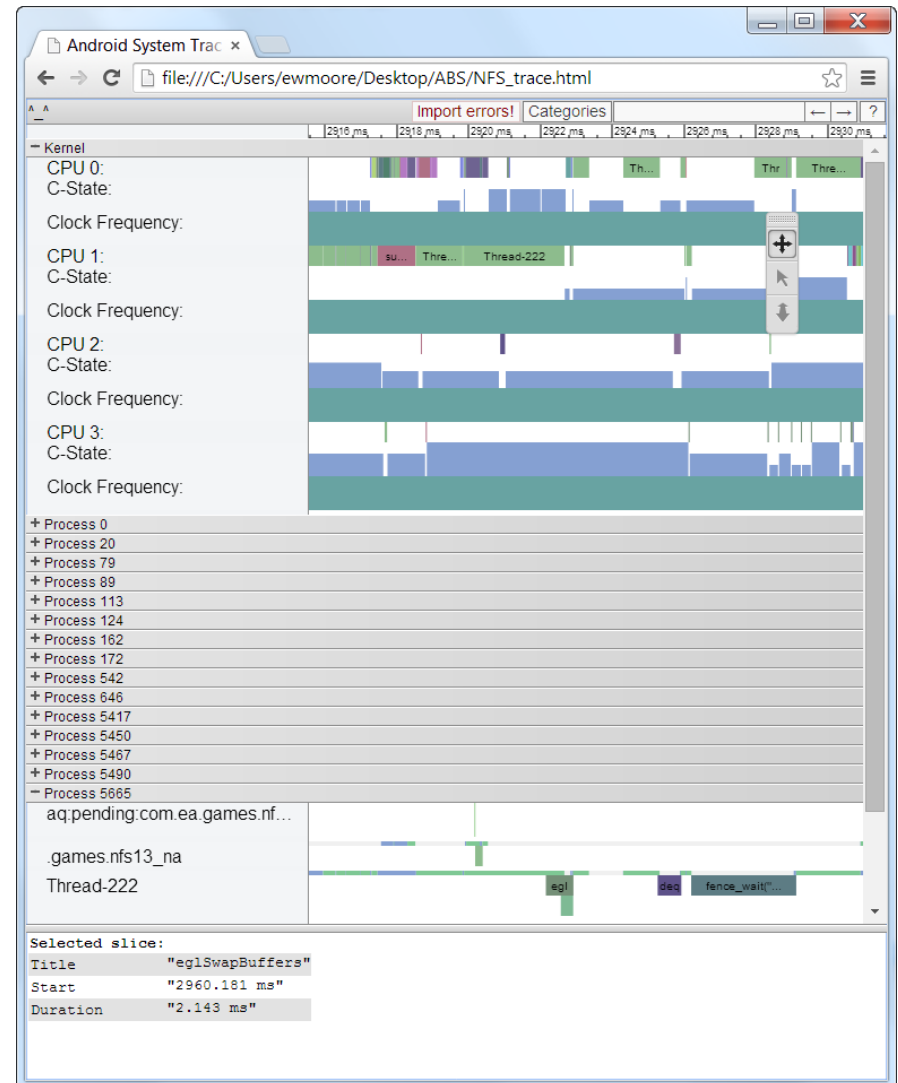
Lots of system information provided.

See System wide what is running

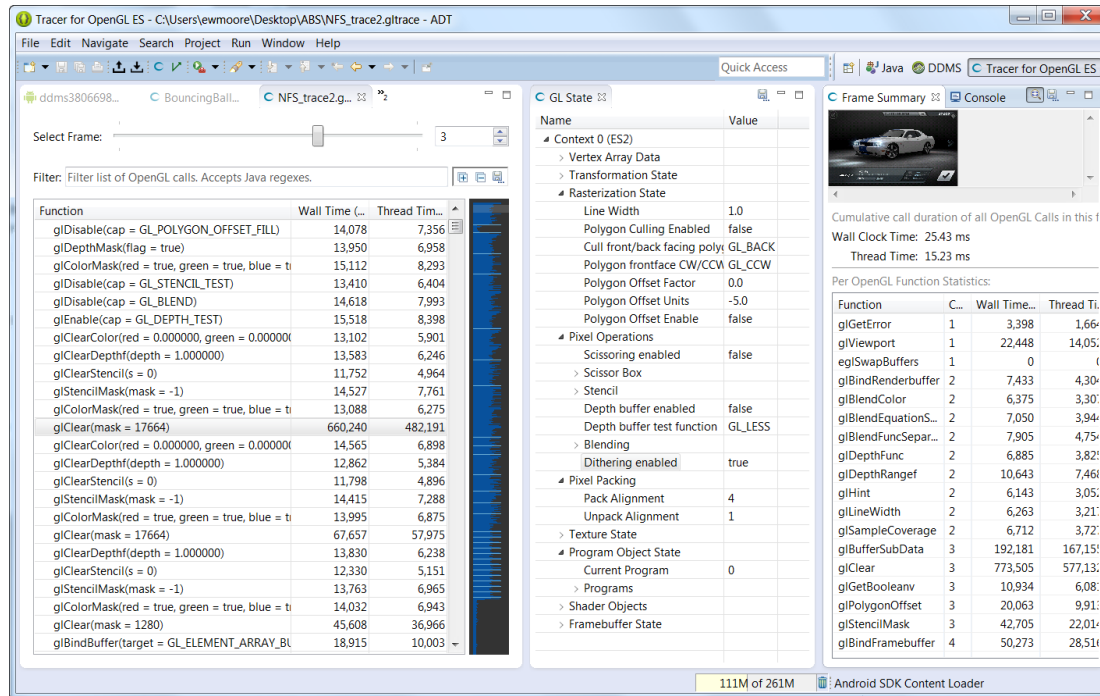
See SurfaceFlinger and vsync

- Can help diagnose frame dropped issues

Can help to determine the code executing during a frame



# OpenGL Tracer



Per OpenGL Draw Call

Get time

Get Context for individual calls

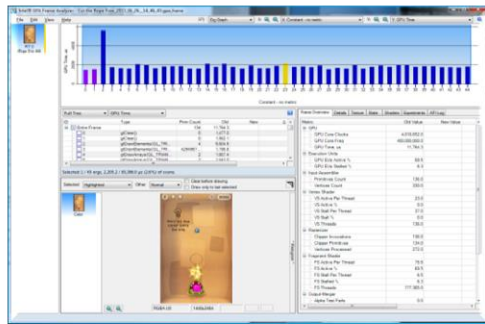
Observer Effect...

Better as a debugging tool

# Intel® GPA Android Support

## System Analyzer

Fast performance analysis with real-time CPU, GPU, OGLES API, and power metrics. GPU Pipeline override experiments allow quick tests to easily identify bottlenecks

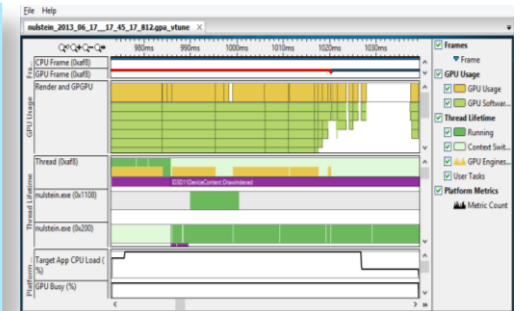


## Frame Analyzer

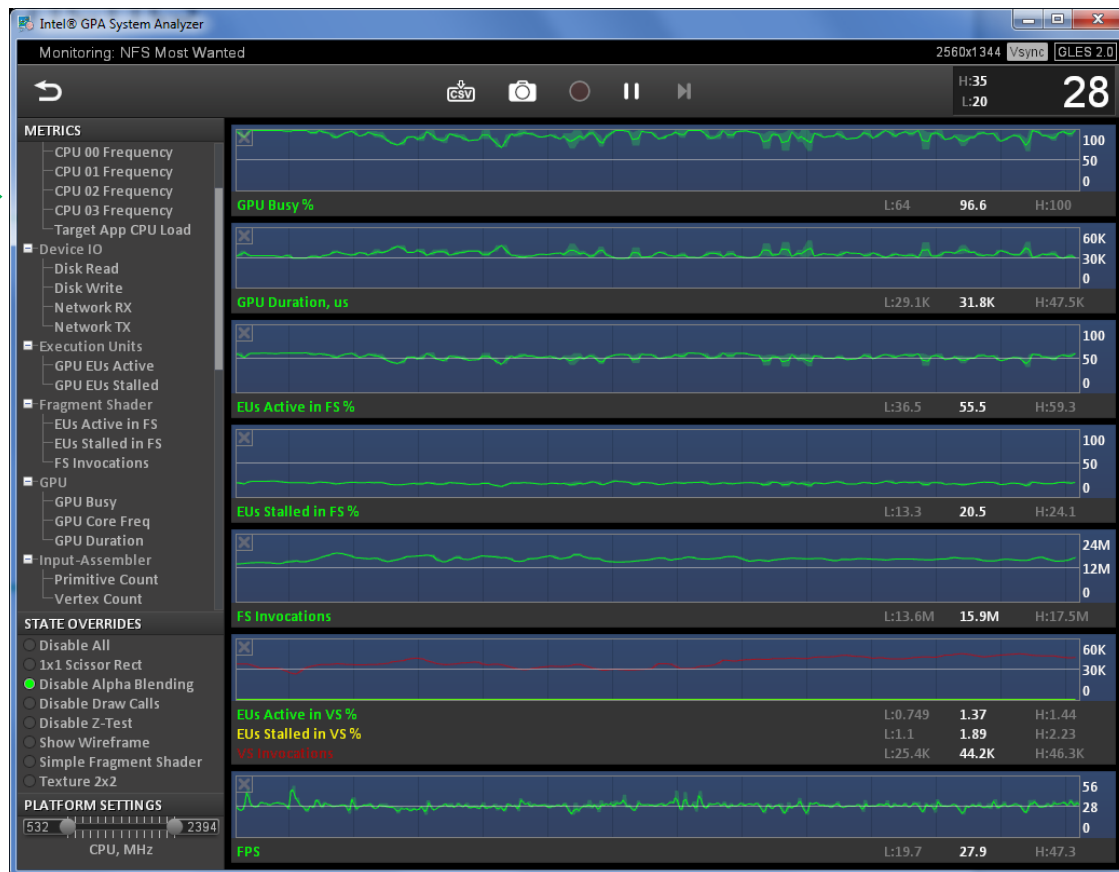
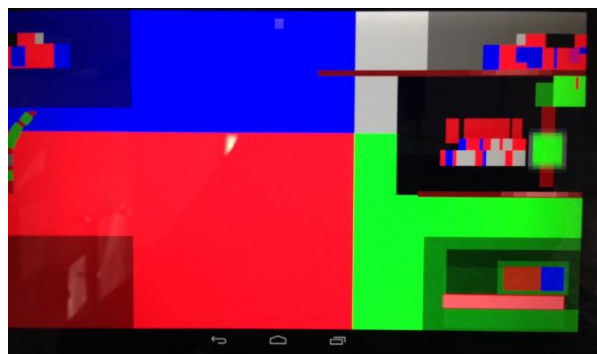
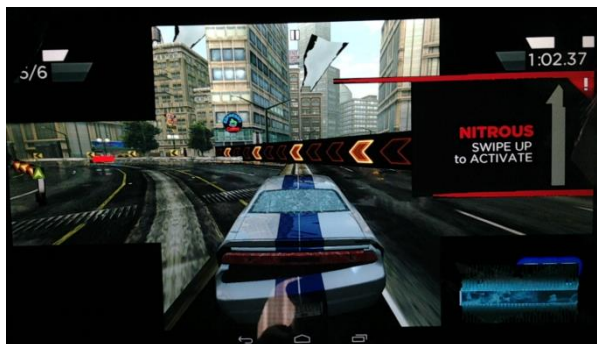
Deep frame performance analysis down to the draw call level, including shaders, textures, Gfx API states, pixel history, and textures.

## New Platform Analyzer

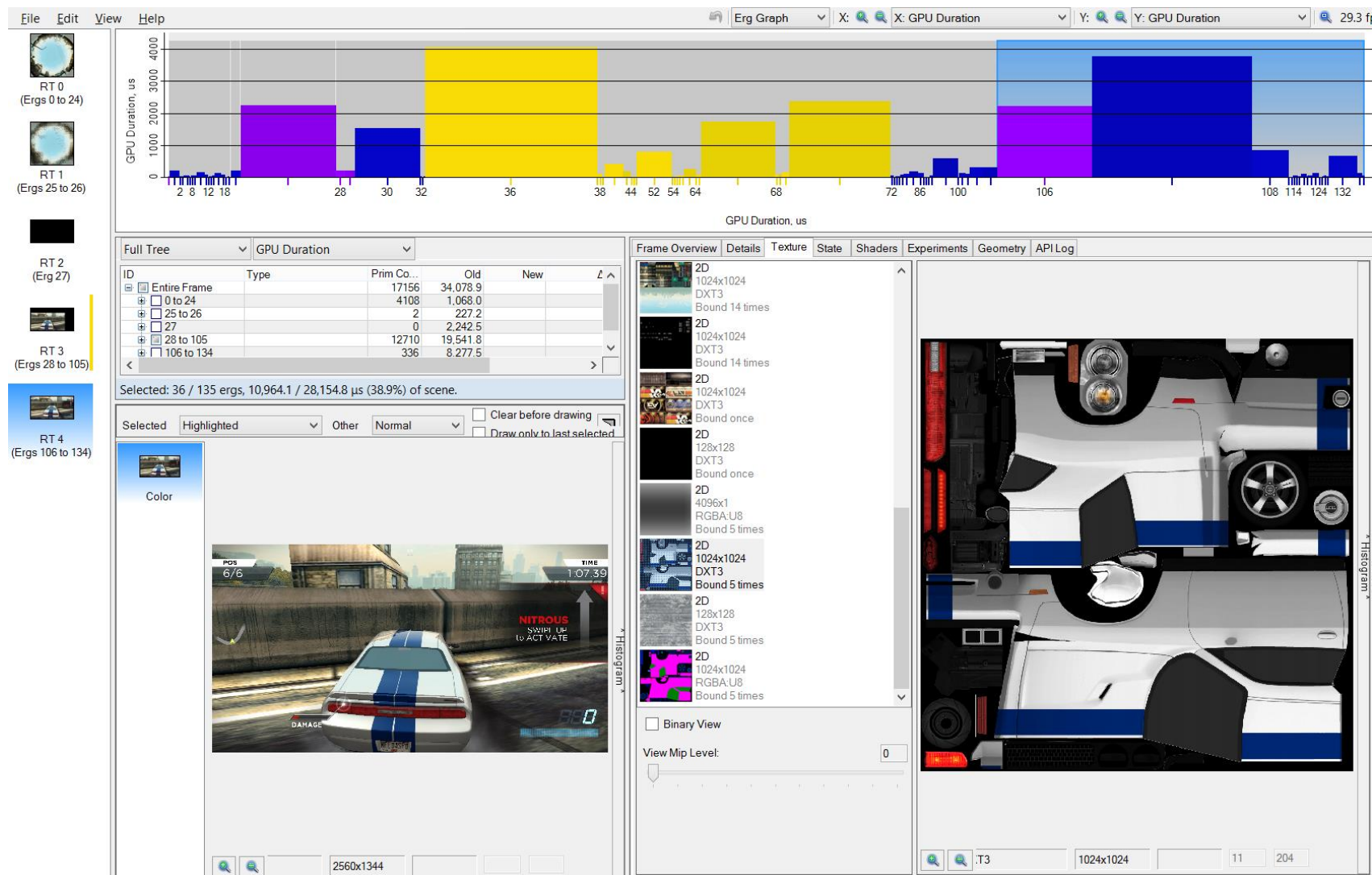
Platform Timeline shows interaction between CPU, GPU, and other platform components. User instrumentation allows detailed task analysis of applications



# System Analyzer – Real time analysis and experiments



# Frame Analyzer - Textures View



# Frame Analyzer - Geometry View

Full Tree ▼ GPU Duration ▼

ID	Type	Prim Co...	Old	New	▲ ▼
[-] Entire Frame		17156	34,078.9		
[-] 0 to 24		4108	1,068.0		
[-] 25 to 26		2	227.2		
[-] 27		0	2,242.5		
[-] 28 to 105		12710	19,541.8		
[-] 106 to 134		336	8,277.5		

Selected: 36 / 135 ergs, 10,964.1 / 28,154.8  $\mu$ s (38.9%) of scene.

Selected Highlighted ▼ Other Normal ▼

Color

2560x1344 2127 78

Frame Overview Details Texture State Shaders Experiments Geometry API Log

Position Index: 0 Show Data Viewer

Erg	Location	X
43	0	12721
44	1	12509
45	2	12569
46	3	12800
47	4	12701
48	5	12507
49	6	12551
50	7	12670
51	8	12509
52	9	12622
53	10	12541
54	11	12721
55	12	12701
56	13	12510
57	14	12661
58	15	12455
59	16	12368

Cull: None Coordinates: Left-handed Mode: Strip Coloring set

Draw call type: glDrawElements Topology: TriList Input primitive count: 2000 Generated primitive count: 2000  
Vertex position format: GL\_SHORT\_3 Index buffer offset: 0 Index buffer size: 6000 Number of vertices referenced: 2317 Extents X: -13171 to 13171 Y: -8964 to 8964 Z: -32750 to 32750 W: 1 to 1



# Frame Analyzer - Editable State and Uniforms

Frame OverviewDetailsTextureStateShadersExperimentsGeometryAPI Log

Render State 2.xRasterization 2.xBlend StateDepth-Stencil StateTexture State 2.xUniforms

2

↓

▲ Multisampling

SampleAlphaToCoverageEnabledFalse

SampleCoverageEnabledFalse

SampleCoverageInvertFalse

SampleCoverageValue1

▲ RasterizerState

CullFaceGL\_BACK

CullFaceEnabled

DitherEnabledTrue

FragmentShaderDerivativeHintOESGL\_DONT\_CARE

FrontFaceGL\_CCW

GenerateMipMapHintGL\_DONT\_CARE

LineWidth1

PolygonOffsetEnabled

PolygonOffsetFactor0

PolygonOffsetUnits

▲ Scissor

ScissorRectangleX=0 Y=0 Width=2560 Height=1344

ScissorTestEnabledFalse

SampleAlphaToCoverageEnabled

If enabled, compute a temporary coverage value where each bit is determined by the alpha value at the corresponding sample location. The temporary coverage value is then ANDed with the fragment coverage value.

Copyright (c) 1991-2006 Silicon Graphics, Inc. Copyright (c) 2010-2013 Khronos Group. This document is licensed under the SGI Free Software B License. For details, see <http://oss.sgi.com/projects/FreeB/>.

# Frame Analyzer - Shader Editor

RT 0  
(Ergs 0 to 24)

RT 1  
(Ergs 25 to 27)

RT 2  
(Erg 27)

RT 3  
(Ergs 28 to 105)

RT 4  
(Ergs 106 to 134)

File Edit View Help

Erg Graph X: GPU Duration Y: GPU Duration 29.3 fp

Full Tree GPU Duration

ID	Type	Prim Co...	Old	New
0	Entire Frame	17156	34,078.9	
0 to 24		4108	1,068.0	
25 to 26		2	227.2	
27		0	2,242.5	
28 to 105		12710	19,541.8	
106 to 134		336	8,277.5	

Selected: 36 / 135 ergs, 10,964.1 / 28,154.8 μs (38.9%) of scene.

Selected Highlighted Other Normal

Color

Frame Overview Details Texture State Shaders Experiments Geometry API Log

Shader	GPU	% GPU	# Ergs
VS.8891	0.0	0.0	1
VS.8898	0.0	0.0	21
VS.8901	0.0	0.0	8
VS.8905	0.0	0.0	1
VS.8909	0.0	0.0	5
FS.8892	0.0	0.0	1
FS.8899	0.0	0.0	21
FS.8902	0.0	0.0	8
FS.8906	0.0	0.0	1
FS.8910	0.0	0.0	5

GLSL

```

uniform highp mat4 sys_Model;
uniform highp float sys_WetRoadTexturePerturbation;
uniform lowp sampler2D FuddleMap;
uniform mediump vec4 sys_CameraPosition;

//Varying
//=====
varying highp vec3 v_2;
varying highp vec2 v_1;
varying highp vec2 v_3;
varying highp vec2 v_4;
varying highp vec3 v_5;
varying highp float v_6;

void main()
{
    highp vec4 t_temp0 = (a_Position0 * a_Position0Scale +
    highp vec4 t_temp1 = (sys_ModelViewProjection * t_temp
    gl_Position = t_temp1;
    v_1 = (a_TexCoord0 * a_TexCoord0Scale + a_TexCoord0Bia
    highp vec4 t_temp2 = (sys_Model * t_temp0);
    v_2 = ((sys_CameraPosition.xyz-t_temp2.xyz));
    v_3 = (a_TexCoord1 * a_TexCoord1Scale + a_TexCoord1Bia
    v_4 = ((t_temp2).xz * sys_RoadSpecularTexCoord);
    v_5 = normalize(mat3(sys_Model[0].xyz,sys_Model[1].xy
    v_6 = (dot(t_temp1.xyw, t_temp1.xyw) * -0.000023);
}
                    
```

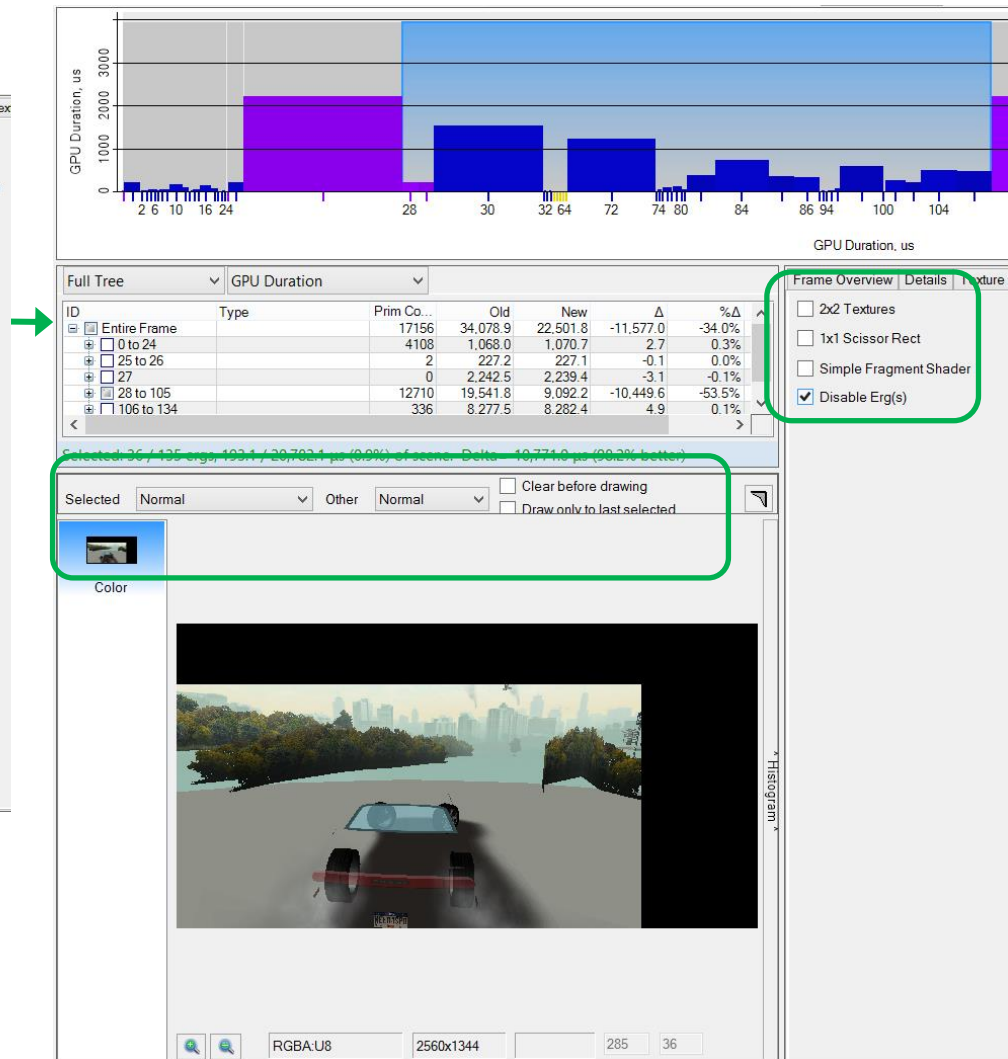
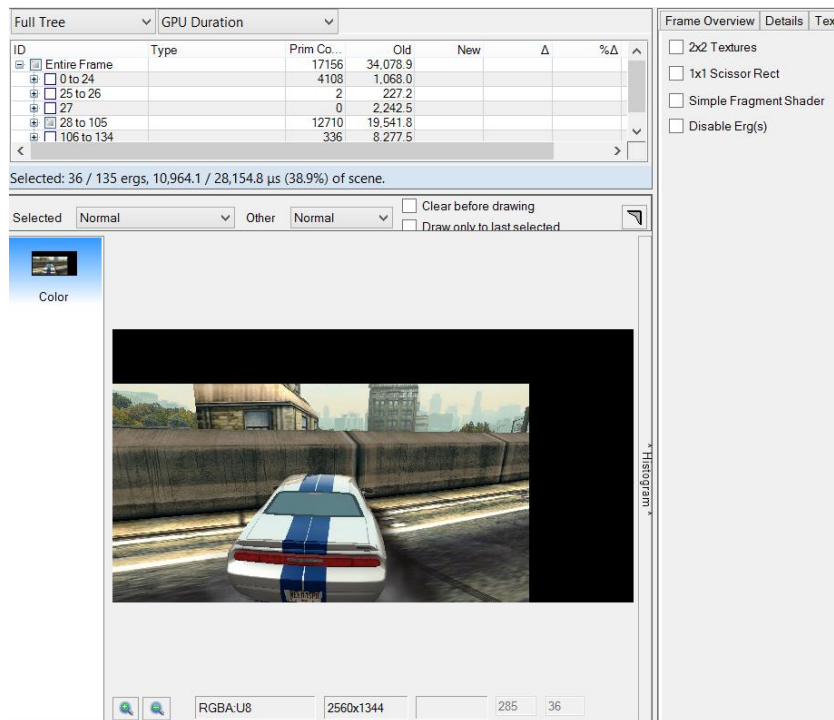
Replace File...

Undo Edits Revert Settings... Apply

2560x1344 2127 78



# Frame Analyzer - Draw Call Performance



# Summary

Performance and Power Methodologies

Showcased various tool options on Android

Google

SysTrace

TraceView

Tracer

Intel® Software Development Tools

Intel® VTune™ 2014 Amplifier for Systems

Intel® Energy Profiler

Intel® Graphics Performance Analyzers

Showed what various system and app issues look like in these tools

# Call to Action – try out the performance tools

## Intel® Native Development Experience at

<http://software.intel.com/en-us/intel-inde> includes

- Intel® Graphics Performance Analyzers
- Android\* SDK and NDK – which includes...
  - TraceView, Systrace, and Tracer

## Intel® System Studio 2014 at

<http://intel.ly/system-studio> includes

- Intel® VTune Amplifier 2014 for Systems
- Intel® System Analyzer

## Get an Intel-based Android\* development device at

<http://software.intel.com/mdk>

<http://01.org/android-ia>

# Additional Resources

# Additional Resources

## <http://intel.ly/system-studio>

<http://software.intel.com/en-us/intel-vtune-amplifier-for-systems>

<http://software.intel.com/en-us/intel-energy-profiler>

Premier Support: <https://premier.intel.com>

Forums: <http://software.intel.com/en-us/forum/intel-system-studio/>

Email: [intelsystemstudio@intel.com](mailto:intelsystemstudio@intel.com)

Release Notes:

[http://software.intel.com/sites/default/files/release\\_notes\\_amplifier\\_for\\_android\\_linux.pdf](http://software.intel.com/sites/default/files/release_notes_amplifier_for_android_linux.pdf)

VTune Amplifier Help Documentation:

[http://software.intel.com/en-us/vtuneampxe\\_2013\\_ug\\_lin](http://software.intel.com/en-us/vtuneampxe_2013_ug_lin)

SubTopic-> [Intel VTune Amplifier User's Guide : Running Analysis Remotely](#)

[http://software.intel.com/sites/default/files/managed/c8/f9/SoCWatchForAndroid\\_v1\\_3\\_0.pdf](http://software.intel.com/sites/default/files/managed/c8/f9/SoCWatchForAndroid_v1_3_0.pdf)

[http://software.intel.com/sites/default/files/managed/9d/59/WakeUpWatch\\_v3\\_1\\_6.pdf](http://software.intel.com/sites/default/files/managed/9d/59/WakeUpWatch_v3_1_6.pdf)

KB Articles: <http://software.intel.com/en-us/articles/intel-system-studio-articles>

<http://software.intel.com/en-us/articles/android-features-in-intel-vtune-amplifier-2014-for-systems-requirements>

<http://software.intel.com/en-us/articles/using-intel-vtune-amplifier-on-non-rooted-android-devices>

<http://software.intel.com/en-us/articles/how-to-use-the-intel-energy-profiler-in-intel-system-studio-2014>

# Intel® System Studio

Deep System Insights for Embedded and Mobile Developers

## Accelerate Time To Market



## Strengthen System Reliability



## Boost Power Efficiency and Performance

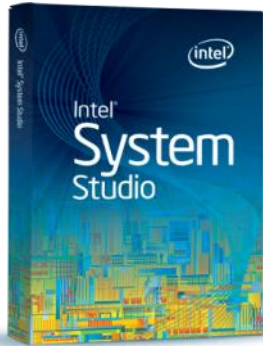


Speed-up development and testing with deep hardware and software insights

Enhance code stability using in-depth system wide debuggers and analyzers

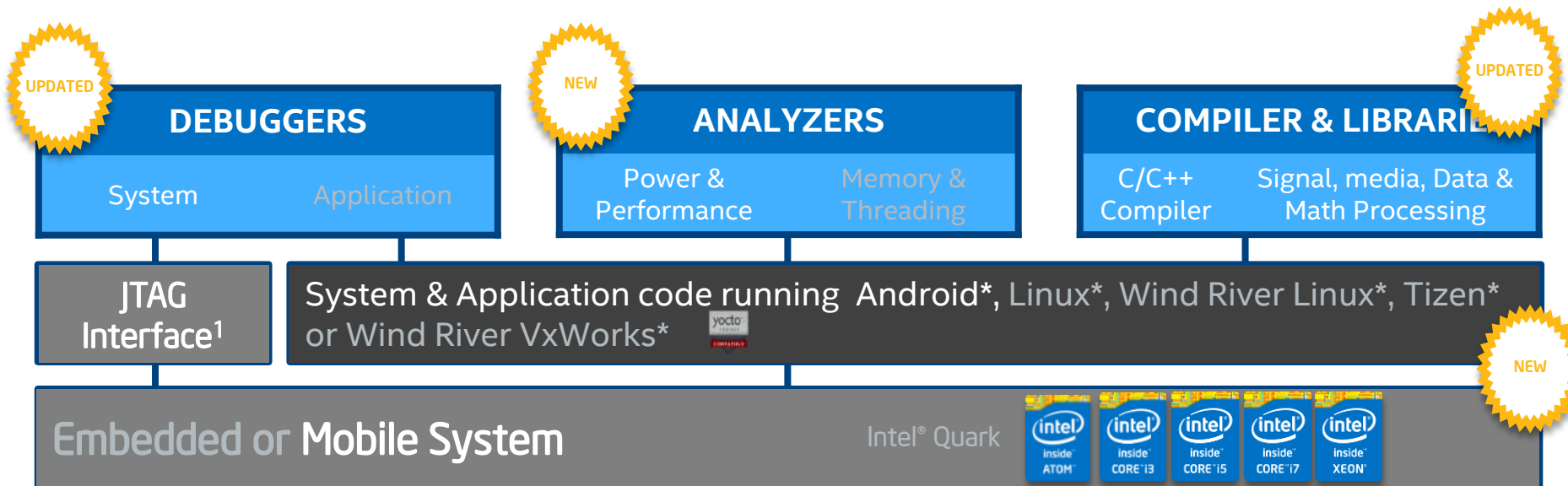
Boost system power efficiency and performance using system-wide analyzers, compilers and libraries

# Intel® System Studio 2014






Integrated software tool suite that provides deep system-wide insights to help:

- Accelerate Time-to-Market
- Strengthen System Reliability
- Boost Power Efficiency and Performance



<sup>1</sup> Optional

# Other Intel® Software Developer Tools for Android\*

1	2	3
As a developer, I care about:	... in these environments:	Developer Solution:
 <p>Writing an app once and having it run anywhere, regardless of OS, device, or architecture.</p>	<ul style="list-style-type: none"> <li>▪ HTML5</li> <li>▪ Cross-OS</li> <li>▪ Cross-platform</li> </ul>	Intel® XDK
 <p>Making my Android® app stand out by delivering native performance that runs on ARM® and runs best on Intel® Architecture-based devices.</p>	<ul style="list-style-type: none"> <li>▪ C++/Java*</li> <li>▪ Intel® Architecture</li> <li>▪ ARM®</li> </ul>	Intel® Integrated Native Developer Experience (Intel® INDE)
 <p>Creating system software including firmware, OS, driver &amp; middleware for dedicated devices</p>	<ul style="list-style-type: none"> <li>▪ C/C++</li> <li>▪ Intel® Architecture</li> </ul>	Intel® System Studio



# Building and Loading the Driver

- Make sure the `KERNEL_SRC_DIR` in `makefile` points to the right kernel of your device
- Do
  - Make `{DISABLE_INTERRUPTS=yes}`
- Driver is built as `timer_compute.ko`
- Move this to the device with `adb push` command
- Load the driver
  - `insmod timer_compute.ko`
- Now for next 25 seconds, we will see timer interrupts every 500ms and some busy work happening

# Plain Video Playback

- Use a video player
- Play a video
- Run it under control of VTune™ and socwatch
  - Collect the data
  - Monitor the framerate
- Visualize the data using VTune™ to get a sense for the baseline performance

# Legal Disclaimer & Optimization Notice

INFORMATION IN THIS DOCUMENT IS PROVIDED "AS IS". NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. INTEL ASSUMES NO LIABILITY WHATSOEVER AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO THIS INFORMATION INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT.

Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests, such as SYSmark and MobileMark, are measured using specific computer systems, components, software, operations and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance tests to assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other products.

Copyright © 2014, Intel Corporation. All rights reserved. Intel, Pentium, Xeon, Xeon Phi, Core, VTune, Cilk, and the Intel logo are trademarks of Intel Corporation in the U.S. and other countries.

## Optimization Notice

Intel's compilers may or may not optimize to the same degree for non-Intel microprocessors for optimizations that are not unique to Intel microprocessors. These optimizations include SSE2, SSE3, and SSSE3 instruction sets and other optimizations. Intel does not guarantee the availability, functionality, or effectiveness of any optimization on microprocessors not manufactured by Intel. Microprocessor-dependent optimizations in this product are intended for use with Intel microprocessors. Certain optimizations not specific to Intel microarchitecture are reserved for Intel microprocessors. Please refer to the applicable product User and Reference Guides for more information regarding the specific instruction sets covered by this notice.

Notice revision #20110804

